HashiCorp

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

NEW QUESTION 1
You have to initialize a Terraform backend before it can be configured.

A. True
B. False

Answer: B

Explanation:
You can configure a backend in your Terraform code before initializing it. Initializing a backend will store the state file remotely and enable features like locking and workspaces. References = [Terraform Backends]

NEW QUESTION 2
Terraform configuration can only import modules from the public registry.

A. True
B. False

Answer: B

Explanation:
Terraform configuration can import modules from various sources, not only from the public registry. Modules can be sourced from local file paths, Git repositories, HTTP URLs, Mercurial repositories, S3 buckets, and GCS buckets. Terraform supports a number of common conventions and syntaxes for specifying module sources, as documented in the [Module Sources] page. References = [Module Sources]

NEW QUESTION 3
terraform validate confirms that your infrastructure matches the Terraform state file.

A. True
B. False

Answer: B

Explanation:
terraform validate does not confirm that your infrastructure matches the Terraform state file. It only checks whether the configuration files in a directory are syntactically valid and internally consistent3. To confirm that your infrastructure matches the Terraform state file, you need to use terraform plan or terraform apply with the -refresh- only option.

NEW QUESTION 4
In Terraform HCL, an object type of object({name=string, age-number}) would match this value.
A)

```
{
    name = "John"
    age = fifty two
}
```

B)

```
{
    name = "John"
    age = 52
}
```

C)

```
{
    name = John
    age = "52"
}
```

D)

```
{
    name = John
    age = fifty two
}
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: B

NEW QUESTION 5
Which is the best way to specify a tag of v1.0.0 when referencing a module stored in Git (for example.
Git::https://example.com/vpc.git)?

A. Append pref=v1.0.0 argument to the source path
B. Add version = ??1.0.0?? parameter to module block
C. Nothing modules stored on GitHub always default to version 1.0.0

Answer: A

Explanation:
 The best way to specify a tag of v1.0.0 when referencing a module stored in Git is to append ?ref=v1.0.0 argument to the source path. This tells Terraform to use a specific Git reference, such as a branch, tag, or commit, when fetching the module source code. For example, source = "git::https://example.com/vpc.git?ref=v1.0.0". This ensures that the module version is consistent and reproducible across different environments. References = [Module Sources], [Module Versions]

NEW QUESTION 6
Which of the following statements about Terraform modules is not true?

A. Modules can call other modules
B. A module is a container for one or more resources
C. Modules must be publicly accessible
D. You can call the same module multiple times

Answer: C

Explanation:
 This is not true, as modules can be either public or private, depending on your needs and preferences. You can use the Terraform Registry to publish and consume public modules, or use Terraform Cloud or Terraform Enterprise to host and manage private modules.

NEW QUESTION 7
Which of these ate features of Terraform Cloud? Choose two correct answers.

A. Automated infrastructure deployment visualization
B. Automatic backups
C. A web-based user interface (UI)
D. Remote state storage

Answer: CD

Explanation:
 These are features of Terraform Cloud, which is a hosted service that provides a web-based UI, remote state storage, remote operations, collaboration features, and more for managing your Terraform infrastructure.

NEW QUESTION 8
You cannot install third party plugins using terraform init.

A. True
B. False

Answer: B

Explanation:
You can install third party plugins using terraform init, as long as you specify the plugin directory in your configuration or as a command-line argument. You can also use the terraform providers mirror command to create a local mirror of providers from any source.

NEW QUESTION 9
If a module declares a variable with a default, that variable must also be defined within the module.

A. True
B. False

Answer: B

Explanation:
A module can declare a variable with a default value without requiring the caller to define it. This allows the module to provide a sensible default behavior that can be customized by the caller if needed. References = [Module Variables]

NEW QUESTION 10
All standard backend types support state locking, and remote operations like plan, apply, and destroy.

A. True
B. False

Answer: B

Explanation:
Not all standard backend types support state locking and remote operations like plan, apply, and destroy. For example, the local backend does not support remote operations and state locking. State locking is a feature that ensures that no two users can make changes to the state file at the same time, which is crucial for preventing race conditions. Remote operations allow running Terraform commands on a remote server, which is supported by some backends like remote or consul, but not all.
References:
? Terraform documentation on backends: Terraform Backends
? Detailed backend support: Terraform Backend Types

NEW QUESTION 10
When you use a remote backend that needs authentication, HashiCorp recommends that you:

A. Write the authentication credentials in the Terraform configuration files
B. Keep the Terraform configuration files in a secret store
C. Push your Terraform configuration to an encrypted git repository
D. Use partial configuration to load the authentication credentials outside of the Terraform code

Answer: D

Explanation:
This is the recommended way to use a remote backend that needs authentication, as it allows you to provide the credentials via environment variables, command-line arguments, or interactive prompts, without storing them in the Terraform configuration files.

NEW QUESTION 14
What are some benefits of using Sentinel with Terraform Cloud/Terraform Cloud? Choose three correct answers.

A. You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0.
B. You can check out and check in cloud access keys
C. Sentinel Policies can be written in HashiCorp Configuration Language (HCL)
D. Policy-as-code can enforce security best practices
E. You can enforce a list of approved AWS AMIs

Answer: ADE

Explanation:
Sentinel is a policy-as-code framework that allows you to define and enforce rules on your Terraform configurations, states, and plans1. Some of the benefits of using Sentinel with Terraform Cloud/Terraform Enterprise are:
•You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0, which would open up your network to the entire internet. This can help you prevent misconfigurations or security vulnerabilities in your infrastructure2.
•Policy-as-code can enforce security best practices, such as requiring encryption, authentication, or compliance standards. This can help you protect your data and meet regulatory requirements3.
•You can enforce a list of approved AWS AMIs, which are pre-configured images that contain the operating system and software you need to run your applications. This can help you ensure consistency, reliability, and performance across your infrastructure4. References =
•1: Terraform and Sentinel | Sentinel | HashiCorp Developer
•2: Terraform Learning Resources: Getting Started with Sentinel in Terraform Cloud
•3: Exploring the Power of HashiCorp Terraform, Sentinel, Terraform Cloud ??

•4: Using New Sentinel Features in Terraform Cloud – Medium

NEW QUESTION 15
The Terraform binary version and provider versions must match each other in a single configuration.

A. True
B. False

Answer: B

Explanation:
The Terraform binary version and provider versions do not have to match each other in a single configuration. Terraform allows you to specify provider version constraints in the configuration??s terraform block, which can be different from the Terraform binary version1. Terraform will use the newest version of the provider that meets the configuration??s version constraints2. You can also use the dependency lock file to ensure Terraform is using the correct provider version3.
References =
•1: Providers - Configuration Language | Terraform | HashiCorp Developer
•2: Multiple provider versions with Terraform - Stack Overflow
•3: Lock and upgrade provider versions | Terraform - HashiCorp Developer

NEW QUESTION 20
What feature stops multiple users from operating on the Terraform state at the same time?

A. State locking
B. Version control
C. Provider constraints
D. Remote backends

Answer: A

Explanation:
State locking prevents other users from modifying the state file while a Terraform operation is in progress. This prevents conflicts and data loss1.

NEW QUESTION 25
You modified your Terraform configuration and run Terraform plan to review the changes. Simultaneously, your teammate manually modified the infrastructure component you are working on. Since you already ran terraform plan locally, the execution plan for terraform apply will be the same.

A. True
B. False

Answer: B

Explanation:
The execution plan for terraform apply will not be the same as the one you ran locally with terraform plan, if your teammate manually modified the infrastructure component you are working on. This is because Terraform will refresh the state file before applying any changes, and will detect any differences between the state and the real resources.

NEW QUESTION 29
You want to define a single input variable to capture configuration values for a server. The values must represent memory as a number, and the server name as a string.
Which variable type could you use for this input?

A. List
B. Object
C. Map
D. Terraform does not support complex input variables of different types

Answer: B

Explanation:
This is the variable type that you could use for this input, as it can store multiple attributes of different types within a single value. The other options are either invalid or incorrect for this use case.

NEW QUESTION 30
You want to define multiple data disks as nested blocks inside the resource block for a virtual machine. What Terraform feature would help you define the blocks using the values in a variable?

A. Local values
B. Count arguments
C. Collection functions
D. Dynamic blocks

Answer: D

Explanation:
Dynamic blocks in Terraform allow you to define multiple nested blocks within a resource based on the values of a variable. This feature is particularly useful for scenarios where the number of nested blocks is not fixed and can change based on variable input.

NEW QUESTION 35
What type of block is used to construct a collection of nested configuration blocks?

A. Dynamic
B. For_each
C. Nesting
D. repeated.

Answer: A

Explanation:
This is the type of block that is used to construct a collection of nested configuration blocks, by using a for_each argument to iterate over a collection value and generate a nested block for each element. For example, you can use a dynamic block to create multiple ingress rules for a security group resource.

NEW QUESTION 40
Which configuration consistency errors does terraform validate report?

A. Terraform module isn't the latest version
B. Differences between local and remote state
C. Declaring a resource identifier more than once
D. A mix of spaces and tabs in configuration files

Answer: C

Explanation:
Terraform validate reports configuration consistency errors, such as declaring a resource identifier more than once. This means that the same resource type and name combination is used for multiple resource blocks, which is not allowed in Terraform. For example, resource "aws_instance" "example" {...} cannot be used more than once in the same configuration. Terraform validate does not report errors related to module versions, state differences, or formatting issues, as these are not relevant for checking the configuration syntax and structure. References = [Validate Configuration], [Resource Syntax]

NEW QUESTION 41
How is terraform import run?

A. As a part of terraform init
B. As a part of terraform plan
C. As a part of terraform refresh
D. By an explicit call
E. All of the above

Answer: D

Explanation:
The terraform import command is not part of any other Terraform workflow. It must be explicitly invoked by the user with the appropriate arguments, such as the resource address and the ID of the existing infrastructure to import. References = [Importing Infrastructure]

NEW QUESTION 43
You are writing a child Terraform module that provisions an AWS instance. You want to reference the IP address returned by the child module in the root configuration. You name the instance resource "main".
Which of these is the correct way to define the output value?
A)

```
output "instance_ip_addr" {
    return aws_instance.main.private_ip
}
```

B)

```
output "aws_instance.instance_ip_addr" {
    return aws_instance.main.private_ip
}
```

C)

```
output "aws_instance.instance_ip_addr" {
    value = ${main.private_ip}
}
```

D)

```
output "instance_ip_addr" {
    value = aws_instance.main.private_ip
}
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: D

NEW QUESTION 44
You have deployed a new webapp with a public IP address on a cloud provider. However, you did not create any outputs for your code. What is the best method to quickly find the IP address of the resource you deployed?

A. In a new folder, use the terraform_remote_state data source to load in the state file, then write an output for each resource that you find the state file
B. Run terraform state list to find the name of the resource, then terraform state show to find the attributes including public IP address
C. Run terraform output ip_address to view the result
D. Run terraform destroy then terraform apply and look for the IP address in stdout

Answer: B

Explanation:
This is a quick way to inspect the state file and find the information you need without modifying anything5. The other options are either incorrect or inefficient.

NEW QUESTION 45
Module variable assignments are inherited from the parent module and you do not need to explicitly set them.

A. True
B. False

Answer: B

Explanation:
Module variable assignments are not inherited from the parent module and you need to explicitly set them using the source argument. This allows you to customize the behavior of each module instance.

NEW QUESTION 49
Which of the following are advantages of using infrastructure as code (IaC) instead of provisioning with a graphical user interface (GUI)? Choose two correct answers.

A. Prevents manual modifications to your resources
B. Lets you version, reuse, and share infrastructure configuration
C. Secures your credentials
D. Provisions the same resources at a lower cost
E. Reduces risk of operator error

Answer: BE

Explanation:
Infrastructure as code (IaC) is a way of managing and provisioning cloud infrastructure using programming techniques instead of manual processes1. IaC has many advantages over using a graphical user interface (GUI) for provisioning infrastructure, such as:
•Versioning: IaC allows you to store your infrastructure configuration in a version control system, such as Git, and track changes over time. This enables you to roll back to previous versions, compare differences, and collaborate with other developers2.
•Reusability: IaC allows you to create reusable modules and templates that can be applied to different environments, such as development, testing, and production. This reduces duplication, improves consistency, and speeds up deployment3.
•Sharing: IaC allows you to share your infrastructure configuration with other developers, teams, or organizations, and leverage existing code from open source repositories or registries. This fosters best practices, innovation, and standardization4.

•Risk reduction: IaC reduces the risk of human error, configuration drift, and security breaches that can occur when provisioning infrastructure manually or using a GUI. IaC also enables you to perform automated testing, validation, and compliance checks on your

infrastructure before deploying it5. References =

•1: What is Infrastructure as Code? Explained for Beginners - freeCodeCamp.org
•2: The benefits of Infrastructure as Code - Microsoft Community Hub
•3: Infrastructure as Code : Best Practices, Benefits & Examples - Spacelift
•4: 5 Benefits of Infrastructure as Code (IaC) for Modern Businesses in the Cloud
•5: The 7 Biggest Benefits of Infrastructure as Code - DuploCloud

NEW QUESTION 50
When should you write Terraform configuration files for existing infrastructure that you want to start managing with Terraform?

A. You can import infrastructure without corresponding Terraform code
B. Terraform will generate the corresponding configuration files for you
C. Before you run terraform Import
D. After you run terraform import

Answer: C

Explanation:
 You need to write Terraform configuration files for the existing infrastructure that you want to import into Terraform, otherwise Terraform will not know how to manage it. The configuration files should match the type and name of the resources that you want to import.

NEW QUESTION 51
Module version is required to reference a module on the Terraform Module Registry.

A. True
B. False

Answer: B

Explanation:
 Module version is optional to reference a module on the Terraform Module Registry. If you omit the version constraint, Terraform will automatically use the latest available version of the module

NEW QUESTION 55
Which are forbidden actions when the terraform state file is locked? Choose three correct answers.

A. Terraform state list
B. Terraform destroy
C. Terraform validate
D. Terraform validate
E. Terraform for
F. Terraform apply

Answer: BCF

Explanation:
 The terraform state file is locked when a Terraform operation that could write state is in progress. This prevents concurrent state operations that could corrupt the state.
The forbidden actions when the state file is locked are those that could write state, such as terraform apply, terraform destroy, terraform refresh, terraform taint, terraform
untaint, terraform import, and terraform state *. The terraform validate command is also forbidden, because it requires an initialized working directory with the state file. The allowed actions when the state file is locked are those that only read state, such as terraform plan, terraform show, terraform output, and terraform console. References = [State Locking] and [Command: validate]

NEW QUESTION 58
When do changes invoked by terraform apply take effect?

A. After Terraform has updated the state file
B. Once the resource provider has fulfilled the request
C. Immediately
D. None of the above are correct

Answer: B

Explanation:
 Changes invoked by terraform apply take effect once the resource provider has fulfilled the request, not after Terraform has updated the state file or immediately. The state file is only a reflection of the real resources, not a source of truth.

NEW QUESTION 63
Once you configure a new Terraform backend with a terraform code block, which command(s) should you use to migrate the state file?

A. terraform destroy, then terraform apply
B. terraform init
C. terraform push
D. terraform apply

Answer: A

Explanation:
This command will initialize the new backend and prompt you to migrate the existing state file to the new location4. The other commands are not relevant for this task.

NEW QUESTION 67
You have multiple team members collaborating on infrastructure as code (IaC) using Terraform, and want to apply formatting standards for readability.
How can you format Terraform HCL (HashiCorp Configuration Language) code according to standard Terraform style convention?

A. Run the terraform fmt command during the code linting phase of your CI/CD process Most Voted
B. Designate one person in each team to review and format everyone's code
C. Manually apply two spaces indentation and align equal sign "=" characters in every Terraform file (*.tf)
D. Write a shell script to transform Terraform files using tools such as AWK, Python, and sed

Answer: A

Explanation:
The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability. Running this command on your configuration files before committing them to source control can help ensure consistency of style between different Terraform codebases, and can also make diffs easier to read. You can also use the -check and -diff options to check if the files are formatted and display the formatting changes respectively2. Running the terraform fmt command during the code linting phase of your CI/CD process can help automate this
process and enforce the formatting standards for your team. References = [Command: fmt]2

NEW QUESTION 72
You have used Terraform lo create an ephemeral development environment in the (loud and are now ready to destroy all the Infrastructure described by your Terraform configuration To be safe, you would like to first see all the infrastructure that Terraform will delete.
Which command should you use to show all of the resources that mil be deleted? Choose two correct answers.

A. Run terraform state rm ??
B. Run terraform show :destroy
C. Run terraform destroy and it will first output all the resource that will be deleted before prompting for approval
D. Run terraform plan .destory

Answer: CD

Explanation:
To see all the resources that Terraform will delete, you can use either of these two commands:
? terraform destroy will show the plan of destruction and ask for your confirmation
before proceeding. You can cancel the command if you do not want to destroy the resources.
? terraform plan -destroy will show the plan of destruction without asking for
confirmation. You can use this command to review the changes before
running terraform destroy. References = : Destroy Infrastructure : Plan Command: Options

NEW QUESTION 74
Which command lets you experiment with terraform expressions?

A. Terraform console
B. Terraform validate
C. Terraform env
D. Terraform test

Answer: A

Explanation:
This is the command that lets you experiment with Terraform expressions, by providing an interactive console that allows you to evaluate expressions and see their results. You can use this command to test your expressions before using them in your configuration files.

NEW QUESTION 76
How can you trigger a run in a Terraform Cloud workspace that is connected to a Version Control System (VCS) repository?

A. Only Terraform Cloud organization owners can set workspace variables on VCS connected workspaces
B. Commit a change to the VCS working directory and branch that the Terraform Cloud workspace is connected to
C. Only Terraform Cloud organization owners can approve plans in VCS connected workspaces
D. Only members of a VCS organization can open a pull request against repositories that are connected to Terraform Cloud workspaces

Answer: B

Explanation:
This will trigger a run in the Terraform Cloud workspace, which will perform a plan and apply operation on the infrastructure defined by the Terraform configuration files in the VCS repository.

NEW QUESTION 79
You ate making changes to existing Terraform code to add some new infrastructure. When is the best time to run terraform validate?

A. After you run terraform apply so you can validate your infrastructure

B. Before you run terraform apply so you can validate your provider credentials
C. Before you run terraform plan so you can validate your code syntax
D. After you run terraform plan so you can validate that your state file is consistent with your infrastructure

Answer: C

Explanation:
This is the best time to run terraform validate, as it will check your code for syntax errors, typos, and missing arguments before you attempt to create a plan. The other options are either incorrect or unnecessary.

NEW QUESTION 83
A developer accidentally launched a VM (virtual machine) outside of the Terraform workflow and ended up with two servers with the same name. They don't know which VM Terraform manages but do have a list of all active VM IDs.
Which of the following methods could you use to discover which instance Terraform manages?

A. Run terraform state list to find the names of all VMs, then run terraform state show for each of them to find which VM ID Terraform manages
B. Update the code to include outputs for the ID of all VMs, then run terraform plan to view the outputs
C. Run terraform taint/code on all the VMs to recreate them
D. Use terraform refresh/code to find out which IDs are already part of state

Answer: A

Explanation:
The terraform state list command lists all resources that are managed by Terraform in the current state file1. The terraform state show command shows the attributes of a single resource in the state file2. By using these two commands, you can compare the VM IDs in your list with the ones in the state file and identify which one is managed by Terraform.

NEW QUESTION 85
Where can Terraform not load a provider from?

A. Plugins directory
B. Provider plugin chance
C. Official HashCrop Distribution on releases.hashcrop.com
D. Source code

Answer: D

Explanation:
This is where Terraform cannot load a provider from, as it requires a compiled binary file that implements the provider protocol. You can load a provider from a plugins directory, a provider plugin cache, or the official HashiCorp distribution on releases.hashicorp.com.

NEW QUESTION 88
Terraform variable names are saved in the state file.

A. True
B. False

Answer: B

Explanation:
Terraform variable names are not saved in the state file, only their values are. The state file only stores the attributes of the resources and data sources that are managed by Terraform, not the variables that are used to configure them.

NEW QUESTION 90
When does Sentinel enforce policy logic during a Terraform Cloud run?

A. Before the plan phase
B. During the plan phase
C. Before the apply phase
D. After the apply phase

Answer: C

Explanation:
Sentinel policies are checked after the plan stage of a Terraform run, but before it can be confirmed or the terraform apply is executed3. This allows you to enforce rules on your infrastructure before it is created or modified.

NEW QUESTION 93
Where in your Terraform configuration do you specify a state backend?

A. The resource block
B. The data source block
C. The terraform block
D. The provider block

Answer: C

Explanation:

In Terraform, the backend configuration, which includes details about where and how state is stored, is specified within the terraform block of your configuration. This block is the correct place to define the backend type and its configuration parameters, such as the location of the state file for a local backend or the bucket details for a remote backend like S3.References = This practice is outlined in Terraform's core documentation, which provides examples and guidelines on how to configure various aspects of Terraform's behavior, including state backends .

NEW QUESTION 94
When should you run terraform init?

A. Every time you run terraform apply
B. Before you start coding a new Terraform project
C. After you run terraform plan for the time in a new terraform project and before you run terraform apply
D. After you start coding a new terraform project and before you run terraform plan for thefirst time.

Answer: D

Explanation:

You should run terraform init after you start coding a new Terraform project and before you run terraform plan for the first time. This command will initialize the working directory by downloading the required providers and modules, creating the initial state file, and performing other necessary tasks. References = : Initialize a Terraform Project

NEW QUESTION 97
If you update the version constraint in your Terraform configuration, Terraform will update your lock file the next time you run terraform Init.

A. True
B. False

Answer: A

Explanation:

If you update the version constraint in your Terraform configuration, Terraform will update your lock file the next time you run terraform init3. This will ensure that you use the same provider versions across different machines and runs.

NEW QUESTION 99
How would you reference the "name???? value of the second instance of this resource?

```
resource "aws_instance" "web" {
  count = 2
  name = "terraform-${count.index}"
}
```

A. aws_instance.web(2),name
B. element(aws_instance.web, 2)
C. aws_instance-web(1)
D. aws_instance_web(1),name
E. Aws_instance,web,* , name

Answer: D

Explanation:

In Terraform, when you use the count meta-argument, you can reference individual instances using an index. The indexing starts at 0, so to reference the "name" value of the second instance, you would use aws_instance.web[1].name. This syntax allows you to access the properties of specific instances in a list generated by the count argument.
References:
? Terraform documentation on count and accessing resource instances: Terraform Count

NEW QUESTION 102
When using multiple configuration of the same Terraform provider, what meta-argument must you include in any non-default provider configurations?

A. Alias
B. Id
C. Depends_on
D. name

Answer: A

Explanation:

This is the meta-argument that you must include in any non-default provider configurations, as it allows you to give a friendly name to the configuration and reference it in other parts of your code. The other options are either invalid or irrelevant for this purpose.

NEW QUESTION 106
What does Terraform use the .terraform.lock.hc1 file for?

A. There is no such file
B. Tracking specific provider dependencies
C. Preventing Terraform runs from occurring
D. Storing references to workspaces which are locked

Answer: B

Explanation:
 The .terraform.lock.hcl file is a new feature in Terraform 0.14 that records the exact versions of each provider used in your configuration. This helps ensure consistent
and reproducible behavior across different machines and runs.

NEW QUESTION 111
You have never used Terraform before and would like to test it out using a shared team account for a cloud provider. The shared team account already contains 15 virtual machines (VM). You develop a Terraform configuration containing one VM. perform terraform apply, and see that your VM was created successfully. What should you do to delete the newly-created VM with Terraform?

A. The Terraform state file contains all 16 VMs in the team accoun
B. Execute terraform destroy and select the newly-created VM.
C. Delete the Terraform state file and execute terraform apply.
D. The Terraform state file only contains the one new V
E. Execute terraform destroy.
F. Delete the VM using the cloud provider console and terraform apply to apply the changes to the Terraform state file.

Answer: C

Explanation:
 This is the best way to delete the newly-created VM with Terraform, as it will only affect the resource that was created by your configuration and state file. The other options are either incorrect or inefficient.

NEW QUESTION 113
One remote backend configuration always maps to a single remote workspace.

A. True
B. False

Answer: A

Explanation:
 The remote backend can work with either a single remote Terraform Cloud workspace, or with multiple similarly-named remote workspaces (like networking-dev and networking-prod). The workspaces block of the backend configuration determines which mode it uses. To use a single remote Terraform Cloud workspace, set workspaces.name to the remote workspace??s full name (like networking-prod). To use multiple remote workspaces, set workspaces.prefix to a prefix used in all of the desired remote workspace names. For example, set prefix = ??networking-?? to use Terraform cloud workspaces with names like networking-dev and networking-prod. This is helpful when mapping multiple Terraform CLI workspaces used in a single Terraform configuration to multiple Terraform Cloud workspaces3. However, one remote backend configuration always maps to a single remote workspace, either by name or by prefix. You cannot use both name and prefix in the same backend configuration, or omit both. Doing so will result in a configuration error3. References = [Backend Type: remote]3

NEW QUESTION 116
Which of the following is not a benefit of adopting infrastructure as code?

A. Versioning
B. A Graphical User Interface
C. Reusability of code
D. Automation

Answer: B

Explanation:
 Infrastructure as Code (IaC) provides several benefits, including the ability to version control infrastructure, reuse code, and automate infrastructure management. However, IaC is typically associated with declarative configuration files and does not inherently provide a graphical user interface (GUI). A GUI is a feature that may be provided by specific tools or platforms built on top of IaC principles but is not a direct benefit of IaC itself1.
References = The benefits of IaC can be verified from the official HashiCorp documentation
on ??What is Infrastructure as Code with Terraform??? provided by HashiCorp Developer1.

NEW QUESTION 119
Which of the following is not true of Terraform providers?

A. An individual person can write a Terraform Provider
B. A community of users can maintain a provider
C. HashiCorp maintains some providers
D. Cloud providers and infrastructure vendors can write, maintain, or collaborate on Terraform
E. providers
F. None of the above

Answer: F

Explanation:
 All of the statements are true of Terraform providers. Terraform providers are plugins that enable Terraform to interact with various APIs and services1. Anyone

can write a Terraform provider, either as an individual or as part of a community2. HashiCorp maintains some providers, such as the AWS, Azure, and Google Cloud providers3. Cloud providers and infrastructure vendors can also write, maintain, or collaborate on Terraform providers, such as the VMware, Oracle, and Alibaba Cloud providers. References =
•1: Providers - Configuration Language | Terraform | HashiCorp Developer
•2: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer
•3: Terraform Registry
•: Terraform Registry

NEW QUESTION 122
A Terraform output that sets the "sensitive" argument to true will not store that value in the state file.

A. True
B. False

Answer: A

Explanation:
A Terraform output that sets the "sensitive" argument to true will store that value in the state file. The purpose of setting sensitive = true is to prevent the value from being displayed in the CLI output during terraform plan and terraform apply, and to mask it in the Terraform UI. However, it does not affect the storage of the value in the state file. Sensitive outputs are still written to the state file to ensure that Terraform can manage resources correctly during subsequent operations.
References:
? Terraform documentation on sensitive outputs: Terraform Output Values

NEW QUESTION 126
Which of the following commands would you use to access all of the attributes and details of a resource managed by Terraform?

A. terraform state list ??provider_type.name??
B. terraform state show ??provider_type.name??
C. terraform get ??provider_type.name??
D. terraform state list

Answer: B

Explanation:
The terraform state show command allows you to access all of the attributes and details of a resource managed by Terraform. You can use the resource address (e.g. provider_type.name) as an argument to show the information about a specific
resource. The terraform state list command only shows the list of resources in the state, not their attributes. The terraform get command downloads and installs modules needed for the configuration. It does not show any information about resources. References = [Command: state show] and [Command: state list]

NEW QUESTION 128
Which provider authentication method prevents credentials from being stored in the state file?

A. Using environment variables
B. Specifying the login credentials in the provider block
C. Setting credentials as Terraform variables
D. None of the above

Answer: D

Explanation:
None of the above methods prevent credentials from being stored in the state file. Terraform stores the provider configuration in the state file, which may include sensitive information such as credentials. This is a potential security risk and should be avoided if possible. To prevent credentials from being stored in the state file, you can use one of the following methods:
? Use environment variables to pass credentials to the provider. This way, the credentials are not part of the provider configuration and are not stored in the state file. However, this method may not work for some providers that require credentials to be set in the provider block.
? Use dynamic credentials to authenticate with your cloud provider. This way,
Terraform Cloud or Enterprise will request temporary credentials from your cloud provider for each run and use them to provision your resources. The credentials are not stored in the state file and are revoked after the run is completed. This method is supported for AWS, Google Cloud Platform, Azure, and
Vault. References = : [Sensitive Values in State] : Authenticate providers with dynamic credentials

NEW QUESTION 129
How do you specify a module??s version when publishing it to the public terraform Module Registry?

A. Configuration it in the module's Terraform code
B. Mention it on the module's configuration page on the Terraform Module Registry
C. The Terraform Module Registry does not support versioning modules
D. Tag a release in the associated repo

Answer: D

Explanation:
This is how you specify a module??s version when publishing it to the public Terraform Module Registry, as it uses the tags from your version control system (such as GitHub or GitLab) to identify module versions. You need to use semantic versioning for your tags, such as v1.0.0.

NEW QUESTION 131
Which of these statements about Terraform Cloud workspaces is false?

A. They have role-based access controls

B. You must use the CLI to switch between workspaces
C. Plans and applies can be triggered via version control system integrations
D. They can securely store cloud credentials

Answer: B

Explanation:

The statement that you must use the CLI to switch between workspaces is false. Terraform Cloud workspaces are different from Terraform CLI workspaces. Terraform Cloud workspaces are required and represent all of the collections of infrastructure in an organization. They are also a major component of role-based access in Terraform Cloud. You can grant individual users and user groups permissions for one or more workspaces that dictate whether they can manage variables, perform runs, etc. You can create, view, and switch between Terraform Cloud workspaces using the Terraform Cloud UI, the Workspaces API, or the Terraform Enterprise Provider5. Terraform CLI workspaces are optional and allow you to create multiple distinct instances of a single configuration within one working directory. They are useful for creating disposable environments for testing or experimenting without affecting your main or production environment. You can create, view, and switch between Terraform CLI workspaces using the terraform workspace command6. The other statements about Terraform Cloud workspaces are true. They have role-based access controls that allow you to assign permissions to users and teams based on their roles and responsibilities. You can create and manage roles using the Teams API or the Terraform Enterprise Provider7. Plans and applies can be triggered via version control system integrations that allow you to link your Terraform Cloud workspaces to your VCS repositories. You can configure VCS settings, webhooks, and branch tracking to automate your Terraform Cloud workflow8. They can securely store cloud credentials as sensitive variables that are encrypted at rest and only decrypted when needed. You can manage variables using the Terraform Cloud UI, the Variables API, or the Terraform Enterprise Provider9. References = [Workspaces]5, [Terraform CLI Workspaces]6, [Teams and Organizations]7, [VCS Integration]8, [Variables]9

NEW QUESTION 136
A Terraform provider is NOT responsible for:

A. Exposing resources and data sources based on an APUI
B. Managing actions to take based on resources differences
C. Understanding API interactions with some service
D. Provisioning infrastructure in multiple

Answer: D

Explanation:

This is not a responsibility of a Terraform provider, as it does not make sense grammatically or logically. A Terraform provider is responsible for exposing resources and data sources based on an API, managing actions to take based on resource differences, and understanding API interactions with some service.

NEW QUESTION 138
......

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questons and Answers in PDF Format

Terraform-Associate-003 Practice Exam Features:

* Terraform-Associate-003 Questions and Answers Updated Frequently

* Terraform-Associate-003 Practice Questions Verified by Expert Senior Certified Staff

* Terraform-Associate-003 Most Realistic Questions that Guarantee you a Pass on Your FirstTry

* Terraform-Associate-003 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

100% Actual & Verified — Instant Download, Please Click
Order The Terraform-Associate-003 Practice Test Here